As illustrated, each position can be represented by a four bit binary string ranging from 0000 to 1111. Each successive segment, then, requires one additional bit to identify a particular code string 54 in that segment. In this example, each code string 54 contains thirty-two bits 58.

Please replace paragraph 47 with the following:

A look-up sting 64 may also be designed such that its instruction portion 66 causes CPU, when executing decompression program 38, to write the index 68 of the look-up string to output memory 42.

Please replace Paragraph 48 with the following:

The tables of Figs. 11-15 are examples only. The selected operations and sizes of library 50 and look-up strings 64 were designed for an ARM processor. In this case, library 50 contains approximately 16,000 code strings 54 each containing thirty-two bits. With a library of this size, as CPU 28 executes decompression program 38, the most accessed code-strings 54 in library 50, if not the entire library 50, naturally migrate to processor cache 46. The same is true for the most repeated instructions of decompression program 38. Consequently, CPU 28 is able to inflate compressed data structure 36 accessing volatile and non-volatile memory as few times as possible. For portable computing systems utilizing an ARM processor, this feature increases the speed at which a program or other data is inflated resulting in better performance and an increased battery life.

Please replace Paragraph 49 with the following:

However, the specifics for any given library 50 and set of look-up strings 64 is dependent upon many factors including the type of CPU 28 involved, the compiler used when writing the compressed program, and the specifics of the program itself. Code strings 54 need not contain 32 bits. Look-up strings 64 can contain any number of bits. Library 50 can be broken into any number of segments 52 with each segment 52 containing any number of positions 56. History cache 40 can contain any number of indexed positions 60. The types operations performed on code strings 54 are not limited to those described above with reference to Figs. 13-15. Depending upon the number of bits 58 in each code string 54 and the number of entries in library 50 and history cache 40, it may be desirable to provide an operation for switching three selected bits rather than one or two. It may be desirable to replace two or more bytes of a given code string 54 rather than just one. Moreover, it may be beneficial in some cases to perform an arithmetic operation in which a number greater than 128 is added to or subtracted from a given code string 54. The particular bits in the instruction 66 of a look-up string 64 that identify any given